

PROJECTION ALGORITHMS FOR NONCONVEX MINIMIZATION WITH APPLICATION TO SPARSE PRINCIPAL COMPONENT ANALYSIS *

WILLIAM W. HAGER[†] AND JIAJIE ZHU[‡]

Abstract. We consider concave minimization problems over nonconvex sets. Optimization problems with this structure arise in sparse principal component analysis. We analyze both a gradient projection algorithm and an approximate Newton algorithm where the Hessian approximation is a multiple of the identity. Convergence results are established. In numerical experiments arising in sparse principal component analysis, it is seen that the performance of the gradient projection algorithm is very similar to that of the truncated power method and the generalized power method. In some cases, the approximate Newton algorithm with a Barzilai-Borwein (BB) Hessian approximation can be substantially faster than the other algorithms, and can converge to a better solution.

Key words. sparse principal component analysis, gradient projection, nonconvex minimization, approximate Newton, Barzilai-Borwein method

1. Introduction. Principal component analysis (PCA) is an extremely popular tool in engineering and statistical analysis. It amounts to computing the singular vectors associated with the largest singular values. In its simplest setting, the rank-one approximation, amounts to solving an optimization problem of the form

$$(1.1) \quad \max\{x^T \Sigma x : x \in \mathbb{R}^n, \quad \|x\| = 1\},$$

where $\Sigma = A^T A$ is the covariance matrix associated with the data matrix $A \in \mathbb{R}^{m \times n}$ and $\|\cdot\|$ is the Euclidean norm. As pointed out in [22], there is no loss of generality in assuming that Σ is positive definite since

$$x^T \Sigma x + \sigma = x^T (\Sigma + \sigma I) x$$

whenever x is feasible in (1.1).

The lack of interpretability has been a major concern in PCA. Sparse PCA partly addresses this problem by constraining the number of nonzero components of the maximizing x in (1.1). Given a positive integer κ , the sparse PCA problem associated with (1.1) is

$$(1.2) \quad \max\{x^T \Sigma x : x \in \mathbb{R}^n, \quad \|x\| = 1, \quad \|x\|_0 \leq \kappa\},$$

*March 28, 2015. The authors gratefully acknowledge support by National Science Foundation under grant 1115568 and by Office of Naval Research under grants N00014-11-1-0068 and N00014-15-1-2048.

[†]hager@ufl.edu, <http://people.clas.ufl.edu/hager/>, PO Box 118105, Department of Mathematics, University of Florida, Gainesville, FL 32611-8105. Phone (352) 294-2308. Fax (352) 392-8357.

[‡]zplusj@ufl.edu, <http://people.clas.ufl.edu/zplusj/> PO Box 118105, Department of Mathematics, University of Florida, Gainesville, FL 32611-8105.

where $\|x\|_0$ denotes the number of nonzero components of x . Due to the sparsity constraint in (1.2), the feasible set is no longer convex, which makes the optimization problem more difficult.

In [8] PCA loadings smaller than a certain tolerance are simply set to zero to produce sparse principal components. More recently, optimization-based approaches have been used to introduce sparsity. For example, in [21] sparsity is achieved using an l_1 relaxation. That is, the problem (1.2) is replaced by

$$(1.3) \quad \max\{x^\top \Sigma x : x \in \mathbb{R}^n, \quad \|x\| = 1, \quad \|x\|_1^2 \leq \kappa\},$$

where $\|x\|_1 = |x_1| + |x_2| + \dots + |x_n|$. The solution of the relaxed problem (1.3) yields an upper bound for the solution of (1.2). In [17] the Rayleigh quotient problem subject to an l_1 -constraint is successively maximized using the authors' SCoTLASS algorithm. In [28], the authors formulate a regression problem and propose numerical algorithms to solve it. Their approach can be applied to large-scale data, but it is computationally expensive. In [10] a new semi-definite relaxation is formulated and a greedy algorithm is developed that computes a full set of good solutions for the target number of non-zero coefficients. With total complexity $O(n^3)$, the algorithm is computationally expensive. Other references related to sparse optimization include [9, 13, 14, 16, 20, 25].

Our work is largely motivated by [18], [22], and [27]. In [18] both l_1 -penalized and l_0 -penalized sparse PCA problems are considered and a generalized power method is developed. The numerical experiments show that their approach outperforms earlier algorithms both in solution quality and in computational speed. Recently, [22] and [27] both consider the l_0 -constrained sparse PCA problem and propose an efficient truncated power method. Their algorithms are equivalent and originated from the classic Frank-Wolfe [11] conditional gradient algorithm.

In this paper, we study both the gradient projection algorithm and an approximate Newton algorithm. Convergence results are established and numerical experiments are given for sparse PCA problems of the form (1.2). The algorithms have the same iteration complexity as the fastest current algorithms. The gradient projection algorithm with unit step size has nearly identical performance as that of ConGradU and Tpower. On the other hand, the approximate Newton algorithm can often converge faster to a better objective value than the other algorithms.

The paper is organized as follows. In Section 2 we analyze the gradient projection algorithm when the constraint set is nonconvex. Section 3 introduces and analyzes the approximate Newton scheme. Section 4 examines the performance of the algorithms in some numerical experiments based on classic examples found in the sparse PCA literature.

Notation. If $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is differentiable, then $\nabla f(x)$ denotes the gradient of f , a row vector, while $g(x)$ denotes the gradient of f arranged as a column vector. The subscript k denotes the iteration number. In particular,

x_k is the k -th iterate in x and $g_k = g(x_k)$. The i -th element of the k -th iterate is denoted x_{ki} . $\|\cdot\|$ denotes the Euclidean norm and $\|\cdot\|_0$ denotes cardinality (number of non-zero elements). If $x \in \mathbb{R}^n$, then the support of \mathbf{x} is the set of indices of nonzeros components:

$$\text{supp}(x) = \{i : x_i \neq 0\}.$$

If $\Omega \subset \mathbb{R}^n$, then $\text{conv}(\Omega)$ is the convex hull of Ω . If $\mathcal{S} \subset \{1, 2, \dots, n\}$, then $x_{\mathcal{S}}$ is the vector obtained by replacing x_i for $i \in \mathcal{S}^c$ by 0. If \mathcal{A} is a set, then \mathcal{A}^c is its complement. If A and $B \in \mathbb{R}^{n \times n}$ are symmetric matrices, then $A \leq B$ means that $A - B$ is negative semidefinite. The standard signum function is defined by

$$\text{sgn}(x) = \begin{cases} +1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases}$$

2. Gradient projection algorithm. Let us consider an optimization problem of the form

$$(2.1) \quad \min\{f(x) : x \in \Omega\},$$

where $\Omega \subset \mathbb{R}^n$ is a nonempty, closed set and $f : \Omega \rightarrow \mathbb{R}$ is differentiable on Ω . Often, the gradient projection algorithm is presented in the context of an optimization problem where the feasible set Ω is convex [3, 4, 12]. Since the feasible set for the sparse PCA problem (1.2) is nonconvex, we will study the gradient projection algorithm for a potentially nonconvex feasible set Ω .

The projection of x onto Ω is defined by

$$P_{\Omega}(x) = \arg \min_{y \in \Omega} \|x - y\|.$$

For the constraint set Ω that arises in sparse PCA, the projection can be expressed as follows:

PROPOSITION 2.1. *For the set*

$$(2.2) \quad \Omega = \{x \in \mathbb{R}^n : \|x\| = 1, \quad \|x\|_0 \leq \kappa\},$$

where κ is a positive integer, we have

$$T_{\kappa}(x)/\|T_{\kappa}(x)\| \in P_{\Omega}(x),$$

where $T_{\kappa}(x)$ is the vector obtained from x by replacing $n - \kappa$ elements of x with smallest magnitude by 0.

Proof. If $y \in \Omega$, then $\|x - y\|^2 = \|x\|^2 + 1 - 2\langle x, y \rangle$. Hence, we have

$$(2.3) \quad P_{\Omega}(x) = \arg \min\{-\langle x, y \rangle : \|y\| = 1, \quad \|y\|_0 \leq \kappa\}.$$

In [22, Prop. 4.3], it is shown that the minimum is attained at $y = T_{\kappa}(x)/\|T_{\kappa}(x)\|$. We include the proof since it is short and we need to refer to it later. Given any set $\mathcal{S} \subset \{1, 2, \dots, n\}$, the solution of the problem

$$\min\{-\langle x, y \rangle : \|y\| = 1, \quad \text{supp}(y) = \mathcal{S}\}$$

is $y = x_{\mathcal{S}}/\|x_{\mathcal{S}}\|$ by the Schwarz inequality, and the corresponding objective value is $-\|x_{\mathcal{S}}\|$, where $x_{\mathcal{S}}$ is the vector obtained by replacing x_i for $i \in \mathcal{S}^c$ by 0. Clearly, the minimum is attained when \mathcal{S} is the set of indices of x associated with the κ absolute largest components. \square

In general, when Ω is closed, the projection exists, although it may not be unique when Ω is nonconvex. If $x_k \in \Omega$ is the current iterate, then in one of the standard implementations of gradient projection algorithm, x_{k+1} is obtained by a line search along the line segment connecting x_k and $P_{\Omega}(x_k - s_k g_k)$, where g_k is the gradient at x_k and $s_k > 0$ is the step size. When Ω is nonconvex, this line segment is not always contained in Ω . Hence, we will focus on gradient projection algorithms of the form

$$(2.4) \quad x_{k+1} \in P_{\Omega}(x_k - s_k g_k).$$

Since Ω is closed, x_{k+1} exists for each k . We first observe that $x_{k+1} - x_k$ always forms an obtuse angle with the gradient, which guarantees descent when f is concave.

LEMMA 2.2. *If $x_k \in \Omega$, then*

$$(2.5) \quad \nabla f(x_k)(y - x_k) \leq 0 \quad \text{for all } y \in P_{\Omega}(x_k - s_k g_k).$$

In particular, for $y = x_{k+1}$ this gives

$$(2.6) \quad \nabla f(x_k)(x_{k+1} - x_k) \leq 0.$$

If f is concave over $\text{conv}(\Omega)$, then $f(x_{k+1}) \leq f(x_k)$.

Proof. If $y \in P_{\Omega}(x_k - s_k g_k)$, then since $P_{\Omega}(x_k - s_k g_k)$ is set of elements in Ω closest to $x_k - s_k g_k$, we have

$$(2.7) \quad \|y - (x_k - s_k g_k)\| \leq \|x_k - (x_k - s_k g_k)\| = s_k \|g_k\|.$$

By the Schwarz inequality and (2.7), it follows that

$$g_k^T(y - (x_k - s_k g_k)) \leq \|g_k\| \|y - (x_k - s_k g_k)\| \leq s_k \|g_k\|^2.$$

We rearrange this inequality to obtain (2.5). If f is concave over $\text{conv}(\Omega)$, then

$$(2.8) \quad f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)(x_{k+1} - x_k).$$

By (2.6), $f(x_{k+1}) \leq f(x_k)$. \square

The following result is well known.

PROPOSITION 2.3. *If $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is concave and $\Omega \subset \mathbb{R}^n$, then*

$$(2.9) \quad \inf\{f(x) | x \in \Omega\} = \inf\{f(x) | x \in \text{conv}(\Omega)\},$$

where the first infimum is attained only when the second infimum is attained. If f is differentiable at $x^ \in \arg \min\{f(x) : x \in \Omega\}$, then*

$$(2.10) \quad \nabla f(x^*)(y - x^*) \geq 0 \quad \text{for all } y \in \text{conv}(\Omega).$$

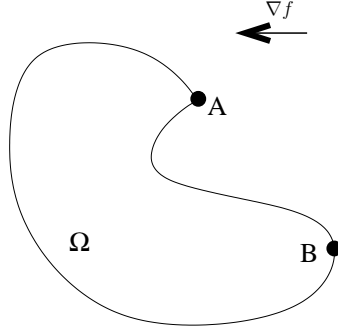


FIG. 2.1. Example that shows Proposition 2.3 may not hold for local minimizers.

Proof. The first result (2.9) is proved in [24, Thm. 32.2]. If x^* minimizes $f(x)$ over Ω , then by (2.9),

$$x^* \in \arg \min\{f(x) : x \in \text{conv}(\Omega)\}.$$

Since $\text{conv}(\Omega)$ is a convex set, the first-order optimality condition for x^* is (2.10). \square

REMARK 1. Note that at a local minimizer x^* of f over a nonconvex set Ω , the inequality $\nabla f(x^*)(y - x^*) \geq 0$ may not hold for all $y \in \Omega$. For example, suppose that $f(x) = a^\top x$ where $\nabla f = a$ has the direction shown in Figure 2.1. The point A is a local minimizer of f over Ω , but (2.10) does not hold. Hence, Proposition 2.3 is only valid for a global minimizer, as stated.

Next, we consider the special choice $y \in P_\Omega(x^* - sg(x^*))$ in Proposition 2.3.

COROLLARY 2.4. If $f : R^n \rightarrow R$ is concave and

$$x^* \in \arg \min\{f(x) : x \in \Omega\},$$

then

$$(2.11) \quad \nabla f(x^*)(y - x^*) = 0$$

whenever $y \in P_\Omega(x^* - sg(x^*))$ for some $s \geq 0$.

Proof. By Proposition 2.3, we have

$$\nabla f(x^*)(y - x^*) \geq 0$$

for all $y \in P_\Omega(x^* - sg(x^*))$. On the other hand, by Lemma 2.2 with $x_k = x^*$, we have

$$\nabla f(x^*)(y - x^*) \leq 0$$

for all $y \in P_\Omega(x^* - sg(x^*))$. Therefore, (2.11) holds. \square

The following property for the projection is needed in the main theorem:

LEMMA 2.5. *If Ω is a nonempty closed set, $x_k \in \mathbb{R}^n$ is a sequence converging to x^* and $y_k \in P_\Omega(x_k)$ is a sequence converging to y^* , then $y^* \in P_\Omega(x^*)$.*

Proof. Since $y_k \in \Omega$ for each k and Ω is closed, $y^* \in \Omega$. Hence, we have

$$\|y^* - x^*\| \geq \min_{y \in \Omega} \|y - x^*\|.$$

If this inequality is a equality, then we are done; consequently, let us suppose that

$$\|y^* - x^*\| > \min_{y \in \Omega} \|y - x^*\| \geq \min_{y \in \Omega} \{\|y - x_k\| - \|x_k - x^*\|\} = \|y_k - x_k\| - \|x_k - x^*\|.$$

As k tends to ∞ , the right side approaches $\|y^* - x^*\|$, which yields a contradiction. \square

We now give further justification for the convergence of the gradient projection algorithm in the nonconvex setting.

THEOREM 2.6. *If $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is concave, Ω is a compact nonempty set, and x_k is generated by the gradient projection algorithm (2.4), then we have $f(x_{k+1}) \leq f(x_k)$ for each k and*

$$(2.12) \quad \lim_{k \rightarrow \infty} \nabla f(x_k)(x_{k+1} - x_k) = 0.$$

If x^ is the limit of any convergent subsequence of the x_k and the step size s_k approaches a limit s^* , then*

$$(2.13) \quad \nabla f(x^*)(y - x^*) \leq 0 \quad \text{for all } y \in P_\Omega(x^* - s^*g(x^*)).$$

If f is continuously differentiable around x^ , then*

$$(2.14) \quad \nabla f(x^*)(y^* - x^*) = 0$$

for some $y^ \in P_\Omega(x^* - s^*g(x^*))$.*

Proof. Sum the concavity inequality (2.8) for $k = 0, 1, \dots, K-1$ to obtain

$$(2.15) \quad f(x_K) - f(x_0) \leq \sum_{k=0}^{K-1} \nabla f(x_k)(x_{k+1} - x_k).$$

Since f is continuous and Ω is compact, $f^* = \min\{f(x) : x \in \Omega\}$ is finite and

$$(2.16) \quad f^* - f(x_0) \leq f(x_K) - f(x_0).$$

Together, (2.15) and (2.16) yield (2.12) since $\nabla f(x_k)(x_{k+1} - x_k) \leq 0$ for each k by Lemma 2.2.

The relation (2.13) is (2.5) with x_k replaced by x^* . For convenience, let x_k also denote the subsequence of the iterates that converges to x^* , and let $y_k \in P_\Omega(x_k - s_k g_k)$ denote the iterate produced by x_k . Since y_k lies in a compact set, there exists a subsequence converging to a limit y^* . Again, for

convenience, let x_k and y_k denote this convergent subsequence. By (2.12) and the fact that f is continuously differentiable around x^* , we have

$$\lim_{k \rightarrow \infty} \nabla f(x_k)(y_k - x_k) = \nabla f(x^*)(y^* - x^*) = 0.$$

By Lemma 2.5, $y^* \in P_\Omega(x^* - s^*g(x^*))$. \square

REMARK 2. *The inequalities (2.6), (2.15), and (2.16) imply that*

$$\min_{0 \leq k \leq K} \nabla f(x_k)(x_k - x_{k+1}) \leq \frac{f(x_0) - f^*}{K+1}.$$

When Ω is convex, much stronger convergence results can be established for the gradient projection algorithm. In this case, the projection onto Ω is unique. By [12, Prop. 2.1], for any $x \in \Omega$ and $s > 0$, $x = P_\Omega(x - sg(x))$ if and only if x is a stationary point for (2.1). That is,

$$\nabla f(x)(y - x) \geq 0 \quad \text{for all } y \in \Omega.$$

Moreover, when Ω is convex,

$$(2.17) \quad \nabla f(x)(P_\Omega(x - sg(x)) - x) \leq -\|P_\Omega(x - \alpha g(x)) - x\|^2/s$$

for any $x \in \Omega$ and $s > 0$. Hence, (2.14) implies that the left side of (2.17) vanishes at $x = x^*$, which means that $x^* = P_\Omega(x^* - sg(x^*))$. And conversely, if $x^* = P_\Omega(x^* - sg(x^*))$, then (2.14) holds.

3. Approximate Newton algorithm. To account for second-order information, Bertsekas [4] analyzes the following version of the gradient projection method:

$$x_{k+1} \in P_\Omega(x_k - s_k \nabla^2 f(x_k)^{-1} g_k).$$

Strong convergence results can be established when Ω is convex and f is strongly convex. On the other hand, if f is concave, local minimizers are extreme points of the feasible set, so the analysis is quite different. Suppose that $\nabla^2 f(x_k)$ is approximated by a multiple α_k of the identity matrix as is done in the BB method [2]. This leads to the approximation

$$(3.1) \quad f(x) \approx f(x_k) + \nabla f(x_k)(x - x_k) + \frac{\alpha_k}{2} \|x - x_k\|^2.$$

Let us consider the algorithm in which the new iterate x_{k+1} is obtained by optimizing the quadratic model:

$$(3.2) \quad x_{k+1} \in \arg \min \{ \nabla f(x_k)(x - x_k) + \frac{\alpha_k}{2} \|x - x_k\|^2 : x \in \Omega \}$$

After completing the square, the iteration is equivalent to

$$x_{k+1} \in \arg \min \{ \alpha_k \|x - (x_k - g_k/\alpha_k)\|^2 : x \in \Omega \}.$$

If $\alpha_k > 0$, then this reduces to $x_{k+1} \in P_\Omega(x_k - g_k/\alpha_k)$; in other words, perform the gradient projection algorithm with step size $1/\alpha_k$. If $\alpha_k < 0$, then the iteration reduces to

$$(3.3) \quad x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k),$$

where

$$(3.4) \quad Q_\Omega(x) = \arg \max\{\|x - y\| : y \in \Omega\}.$$

If Ω is unbounded, then this iteration may not make sense since the maximum could occur at infinity. But if Ω is bounded, then the iteration is justified in the sense that it is based on a quadratic model of the function, which could be better than a linear model. In the special case where Ω is the constraint set (2.2) appearing in sparse PCA and $\alpha_k < 0$, the maximization in (3.4) can be evaluated as follows:

PROPOSITION 3.1. *For the set Ω in (2.2) associated with sparse PCA, we have*

$$-T_\kappa(x)/\|T_\kappa(x)\| \in Q_\Omega(x).$$

Proof. As in the proof of Proposition 2.1, $\|x - y\|^2 = \|x\|^2 + 1 - 2\langle x, y \rangle$ when y lies in the set Ω of (2.2). Hence, we have

$$Q_\Omega(x) = \arg \max\{-\langle x, y \rangle : \|y\| = 1, \quad \|y\|_0 \leq \kappa\}.$$

Given any set $\mathcal{S} \subset \{1, 2, \dots, n\}$, the solution of the problem

$$\max\{-\langle x, y \rangle : \|y\| = 1, \quad \text{supp}(y) = \mathcal{S}\}$$

is $y = -x_{\mathcal{S}}/\|x_{\mathcal{S}}\|$ by the Schwarz inequality, and the corresponding objective value is $\|x_{\mathcal{S}}\|$. Clearly, the maximum is attained when \mathcal{S} corresponds to a set of indices of x associated with the κ absolute largest components. \square

Let us now study the iterates generated by the quadratic model (3.2). If $\alpha_k > 0$, then the iteration reduces to $x_{k+1} \in P_\Omega(x_k - g_k/\alpha_k)$, which was studied in Section 2. Hence, we focus on the case $\alpha_k < 0$ where the iteration reduces to $x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k)$.

LEMMA 3.2. *If $x_k \in \Omega$ and $\alpha_k < 0$, then*

$$(3.5) \quad \nabla f(x_k)(y - x_k) + \frac{\alpha_k}{2}\|y - x_k\|^2 \leq 0 \quad \text{for all } y \in Q_\Omega(x_k - g_k/\alpha_k).$$

In particular, for $y = x_{k+1}$ this gives

$$(3.6) \quad \nabla f(x_k)(x_{k+1} - x_k) + \frac{\alpha_k}{2}\|x_{k+1} - x_k\|^2 \leq 0.$$

If f satisfies

$$(3.7) \quad f(y) \leq f(x) + \nabla f(x)(y - x) + \frac{\alpha_k}{2}\|y - x\|^2 \quad \text{for all } x \text{ and } y \in \Omega,$$

then $f(x_{k+1}) \leq f(x_k)$.

Proof. If $y \in Q_\Omega(x_k - g_k/\alpha_k)$, then since $Q_\Omega(x_k - s_k g_k)$ is set of elements in Ω farthest from $x_k - g_k/\alpha_k$, we have

$$(3.8) \quad \|y - (x_k - g_k/\alpha_k)\|^2 \geq \|x_k - (x_k - g_k/\alpha_k)\|^2 = \|g_k\|^2/\alpha_k^2.$$

We can square and rearrange the left side of (3.8) to obtain (3.5). If (3.7) holds, then we combine this with (3.6) to obtain $f(x_{k+1}) \leq f(x_k)$. \square

The condition (3.7) is satisfied if f is twice continuously differentiable over $\text{conv}(\Omega)$ and α_k exceeds the largest eigenvalue of the Hessian. We now show that under suitable assumptions, convergent subsequences of the iterates $x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k)$ approach a point where the first-order optimality condition (2.3) for a global optimizer holds.

THEOREM 3.3. *Suppose that Ω is compact, f is continuously differentiable on Ω , x^* is a limit of any convergent subsequence of the iterates generated by the scheme $x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k)$, where $\alpha_k < 0$ for each k . If $\|x_{k+1} - x_k\|$ tends to 0, then*

$$(3.9) \quad \nabla f(x^*)(y - x^*) \geq 0 \quad \text{for all } y \in \text{conv}(\Omega).$$

Proof. The quadratic model (3.1) is concave when $\alpha_k < 0$. Hence, by Proposition 2.3,

$$(3.10) \quad [\nabla f(x_k) + \alpha_k(x_{k+1} - x_k)^\top](y - x_{k+1}) \geq 0$$

for all $y \in \text{conv}(\Omega)$. Let us focus on a subsequence of the x_k converging to x^* . Since $\|x_{k+1} - x_k\|$ tends to 0, those x_{k+1} associated with the convergent x_k subsequence also converge to x^* . In the limit, (3.10) yields (3.9). \square

When f is strongly concave, a line search strategy can be used to ensure that $\|x_{k+1} - x_k\|$ tends to 0 and hence, the optimality condition (3.9) holds at all limit points. For illustration, consider the following algorithm:

MONOTONE APPROXIMATE NEWTON (FOR STRONGLY CONCAVE f)

Given $\sigma \in (0, 1)$, $[\alpha_{\min}, \alpha_{\max}] \subset (-\infty, 0)$, and starting guess \mathbf{x}_1 .

Set $k = 1$.

Step 1. Choose $\beta_k \in [\alpha_{\min}, \alpha_{\max}]$

Step 2. Set $\alpha_k = \sigma^j \beta_k$ where $j \geq 0$ is the smallest integer such that

$$f(x_{k+1}) \leq f(x_k) + \alpha_k \|x_{k+1} - x_k\|^2 \quad \text{where}$$

$$x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k)$$

Step 3. If a stopping criterion is satisfied, terminate.

Step 4. Set $k = k + 1$ and go to step 1.

The objective values generated by this algorithm are monotone nonincreasing since $\alpha_k < 0$. If Ω is compact, then f is bounded from below and $f(x_k) - f(x_{k+1})$ approaches 0. It follows from Step 2 that

$$\|x_{k+1} - x_k\|^2 \leq \frac{f(x_k) - f(x_{k+1})}{-\alpha_k}.$$

We now show that when f is strongly concave, α_k is bounded away from 0, uniformly in k . Hence, $\|x_{k+1} - x_k\|$ tends to 0.

LEMMA 3.4. *If f is differentiable on Ω and for some $\mu < 0$, we have*

$$(3.11) \quad f(y) \leq f(x) + \nabla f(x)(y - x) + \frac{\mu}{2}\|y - x\|^2 \quad \text{for all } x \text{ and } y \in \Omega,$$

then Step 2 in the monotone approximate Newton algorithm terminates with a finite j , and α_k bounded away from 0, uniformly in k .

Proof. If $x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k)$, then by (3.11) with $y = x_{k+1}$ and $x = x_k$ and by (3.5), we have

$$(3.12) \quad f(x_{k+1}) \leq f(x_k) + \left(\frac{\mu - \alpha_k}{2}\right)\|x_{k+1} - x_k\|^2.$$

If $0 > \alpha_k > \mu/2$, then $\mu - \alpha_k \leq \alpha_k$. Hence, by (3.12), Step 2 must terminate whenever $\alpha_k > \mu/2$. Since $\sigma \in (0, 1)$, it follows that Step 2 terminates for a finite j . If Step 2 terminates when $j > 0$, then $\sigma^{j-1}\beta \leq \mu/2$, which implies that $\alpha_k = \sigma^j\beta \leq \sigma\mu/2$. If Step 2 terminates when $j = 0$, then $\alpha_k \leq \alpha_{\max} < 0$. In either case, α_k is uniformly bounded away from 0. \square

One way to choose the Hessian approximation α_k in (3.3) or β_k in the Monotone Approximate Newton Algorithm is with the BB approximation [2] given by

$$(3.13) \quad \alpha_k^{BB} = \frac{\|\nabla f(x_k) - \nabla f(x_{k-1})\|^2}{\|x_k - x_{k-1}\|^2}.$$

4. Numerical experiments. We will investigate the performance of the gradient project and approximate Newton algorithm relative to previously developed algorithms in the literature. In our experiments we use the gradient projection algorithm with unit stepsize (GPU):

$$x_{k+1} \in P_\Omega(x_k - g_k).$$

And in our experiments with the approximate Newton algorithm, we employ the BB approximation (GPBB):

$$x_{k+1} \in Q_\Omega(x_k - g_k/\alpha_k^{BB}).$$

The performance of this nonmonotone scheme will be compared later to that of the Monotone Approximate Newton Algorithm. For the set Ω associated with sparse PCA, we have

$$T_\kappa(x)/\|T_\kappa(x)\| \in P_\Omega(x) \quad \text{and} \quad -T_\kappa(x)/\|T_\kappa(x)\| \in Q_\Omega(x)$$

by Propositions 2.1 and 3.1 respectively.

We compare the performance of our algorithms to those of both the truncated power method (Tpower) [27] and the generalized power method (Gpower) [18]. The conditional gradient algorithm with unit step-size (ConGradU) proposed in [22] is equivalent to the truncated power method. Both truncated and generalized power method are targeted to the sparse PCA problem (1.2). The truncated power method handles the sparsity constraint by pushing the absolute smallest components of the iterates to 0. The iteration can be expressed

$$(4.1) \quad x_{k+1} = \frac{T_\kappa(-g_k)}{\|T_\kappa(-g_k)\|}.$$

For comparison, an iteration of the gradient projection algorithm with unit step size GPU is given by

$$(4.2) \quad x_{k+1} = \frac{T_\kappa(x_k - g_k)}{\|T_\kappa(x_k - g_k)\|},$$

while the approximate Newton algorithm is

$$(4.3) \quad x_{k+1} = \text{sgn}(\alpha_k) \frac{T_\kappa(x_k - g_k/\alpha_k^{BB})}{\|T_\kappa(x_k - g_k\alpha_k^{BB})\|}.$$

Since the computation of α_k^{BB} requires the gradient at two points, we start GPBB with one iteration of GPU. For the sparse PCA problem (1.2), the time for one iteration of any of these methods is basically the time to multiply a vector by the covariance matrix Σ . Note that the monotone approximate Newton algorithm could be more costly since the evaluation of an acceptable j may require several evaluations of the objective function.

In the generalized power method, the sparsity constraint is handled using a penalty terms. If $\gamma > 0$ denotes the penalty, then Gpower_{l_1} corresponds to the optimization problem

$$\max_{\|x\|=1} \sqrt{x^\top \Sigma x} - \gamma \|x\|_1,$$

where $\|x\|_1 = |x_1| + |x_2| + \dots + |x_n|$, while Gpower_{l_0} corresponds to

$$\max_{\|x\|=1} x^\top \Sigma x - \gamma \|x\|_0.$$

The parameter γ needs to be tuned to achieve the desired cardinality; as γ increases, the cardinality of the Gpower approximation decreases. In contrast, the cardinality is an explicit input parameter for either the truncated power method or for our algorithms; in many applications, cardinality is often specified.

All experiments were conducted using MATLAB on a GNU/Linux computer with 8GB of RAM and an Intel Core i7-2600 processor. For the starting

guess in our experiments, we follow the practice of the Tpower algorithm [27] and set $x = e_i$, the i -th column of the identity matrix, where i is the index of the largest diagonal element of the covariance matrix Σ . Our numerical experiments are based on the sparse PCA problem (1.2). We measure the quality of the solution to (1.2) computed by any of the methods using the ratio $x^\top \Sigma x / y^\top \Sigma y$ where x is the sparse first principal component computed by any of the algorithms for (1.2) and y is the first principal component (a solution of (1.1)). This ratio is often called the *proportion of the explained variance*.

Pit props dataset. This dataset [15] contains 180 observations with 13 variables, and a covariance matrix $\Sigma \in \mathbb{R}^{13 \times 13}$. This is a standard benchmark dataset for Sparse PCA algorithms. We consider $\kappa = 6$ or 7, and we adjust the value of γ to achieve the same sparsity in Gpower. The last column of Table 4.1 gives the proportion of the explained variance. Observe that all the methods achieve essentially the same proportion of the explained variance.

TABLE 4.1
Results on Pit props data set.

Method	Parameters	Explained Variance
GPBB	$\kappa = 6$	0.8939
GPBB	$\kappa = 7$	0.9473
GPU	$\kappa = 6$	0.8939
GPU	$\kappa = 7$	0.9473
Tpower(ConGradU)	$\kappa = 6$	0.8939
Tpower(ConGradU)	$\kappa = 7$	0.9473
Gpower $_{l_1}$	$\gamma = 0.5 (\Leftrightarrow \kappa = 6)$	0.8939
Gpower $_{l_1}$	$\gamma = 0.4 (\Leftrightarrow \kappa = 7)$	0.9473
Gpower $_{l_0}$	$\gamma = 0.2 (\Leftrightarrow \kappa = 6)$	0.8939
Gpower $_{l_0}$	$\gamma = 0.15 (\Leftrightarrow \kappa = 7)$	0.9473

We also considered multiple sparse principal components for this data set and got the same results as those obtained in Table 2 of [28] for the Tpower and PathSPCA algorithms. Similarly, for the lymphoma data set [1] and the Ramaswamy data set [23], all methods yielded the same proportion of explained variance, although the value of γ for Gpower needed to be tuned to achieve the specified cardinality.

Randomly generated data. In the next set of experiments, we consider randomly generated data, where $\Sigma = A^\top A$ with each entry of $A \in \mathbb{R}^{m \times n}$ generated by a normal distribution with mean 0 and standard deviation 1. For randomly generated matrices, we can study the performance as either m or κ change. Each result that we present is based on an average over 100

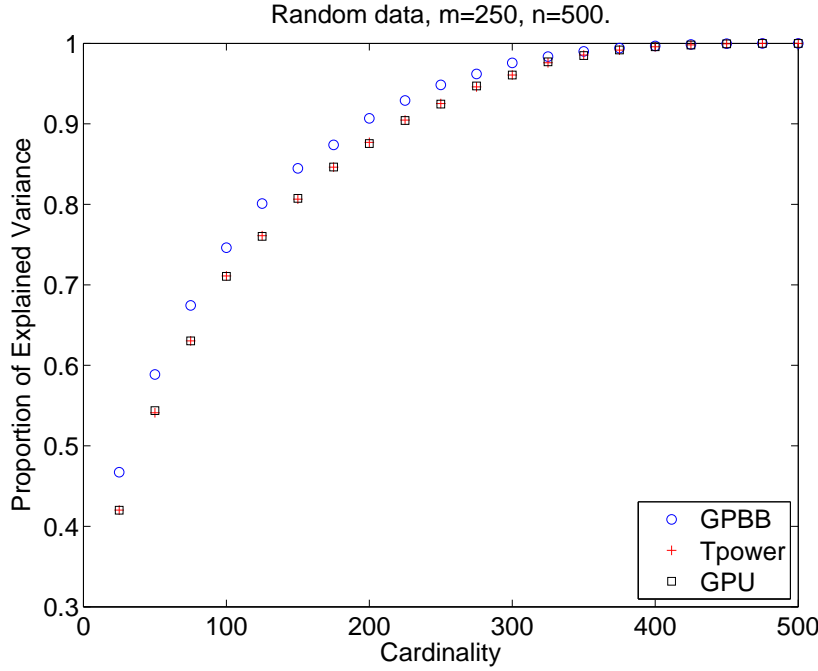


FIG. 4.1. *Explained variance versus cardinality for random data set.*

randomly generated matrices. In Figure 4.1 we plot the proportion of the explained variance versus cardinality for $m = 250$ and $n = 500$. Observe that GPBB yields a significantly better objective value as the cardinality decreases when compared to either GPU or ConGradU, while GPU and ConGradU have essentially identical performance. Even though all the algorithms seem to yield similar results in Figure 4.1 as the cardinality approaches 500, the convergence of the algorithms is quite different. To illustrate this, let us consider the case where the cardinality is 500. In this case where $\kappa = n$, the sparse PCA problem (1.2) and the original PCA problem (1.1) are identical, and the solution of (1.1) is a normalized eigenvector associated with the largest eigenvalue λ_1 of Σ . Since the optimal objective value is known, we can compute the relative error

$$\frac{|\lambda_1^{\text{exact}} - \lambda_1^{\text{approx}}|}{|\lambda_1^{\text{exact}}|},$$

where $\lambda_1^{\text{approx}}$ is the approximation to the optimal objective value generated by any of the algorithms. In Figure 4.2 we plot the base 10 logarithm of the relative error versus the iteration number. Observe that GPBB is able to reduce the relative error to the machine precision near 10^{-16} in about 175 iterations, while ConGradU and GPU have relative error around 10^{-3} after 200 iterations. To achieve a relative error around 10^{-16} , ConGradU and GPU

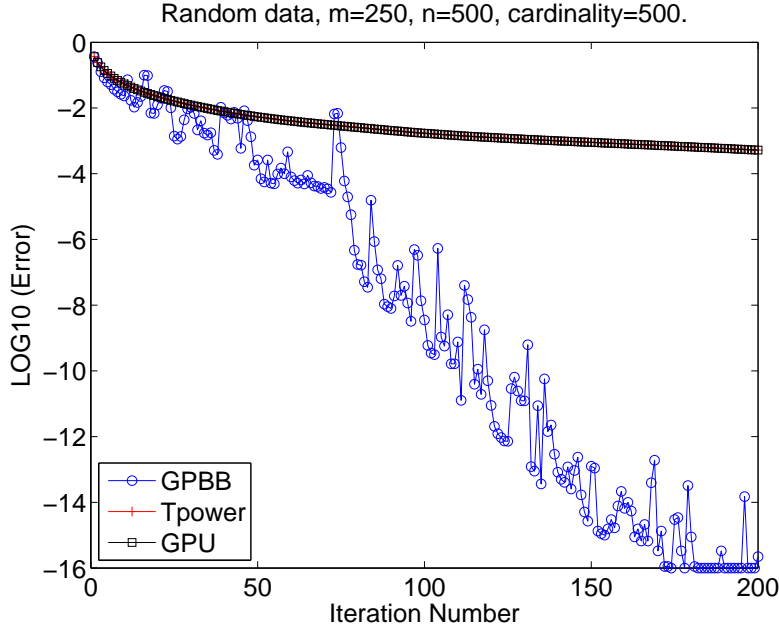


FIG. 4.2. A plot of the base 10 logarithm of the relative error versus iteration number for the random data set with $m = 250$ and cardinality $\kappa = 500$.

require about 4500 iterations, roughly 25 times more than GPBB.

In this example, the condition (3.7) of Lemma 3.2 is not satisfied and the objective values produced by GPBB do not improve monotonically. Nonetheless, the convergence is relatively fast. The results for the explained variance in Figure 4.1 were obtained by running either ConGradU or GPU for 6000 iterations, while GPBB was run for 200 iterations. Hence, the better objective values obtained by GPBB in Figure 4.1 were due to the algorithm converging to a better solution, rather than to premature termination of either GPU or ConGradU.

In Figure 4.3 we plot the proportion of the explained variance versus the iteration number when $m = 250$, $n = 500$, and the cardinality $\kappa = 50$ in the random data set. When we plot the function value as in Figure 4.3, it is more difficult to see the nonmonotone nature of the convergence for GPBB. This nonmonotone nature is clearly visible in Figure 4.2 where we plot the error instead of the function value. In Figure 4.4 we show how the explained variance depends on m . As m increases, the explained variance associated with GPBB becomes much better than that of either ConGradU or GPU.

As mentioned already, the nonmonotone convergence of GPBB is due to the fact that the BB value for α_k may not satisfy condition (3.7). We can achieve monotone convergence by increasing α_k until it is large enough that $f(x_{k+1}) \leq f(x_k)$ as in the monotone approximate Newton algorithm. In Figure 4.5 we compare the relative error for the monotone scheme for the non-

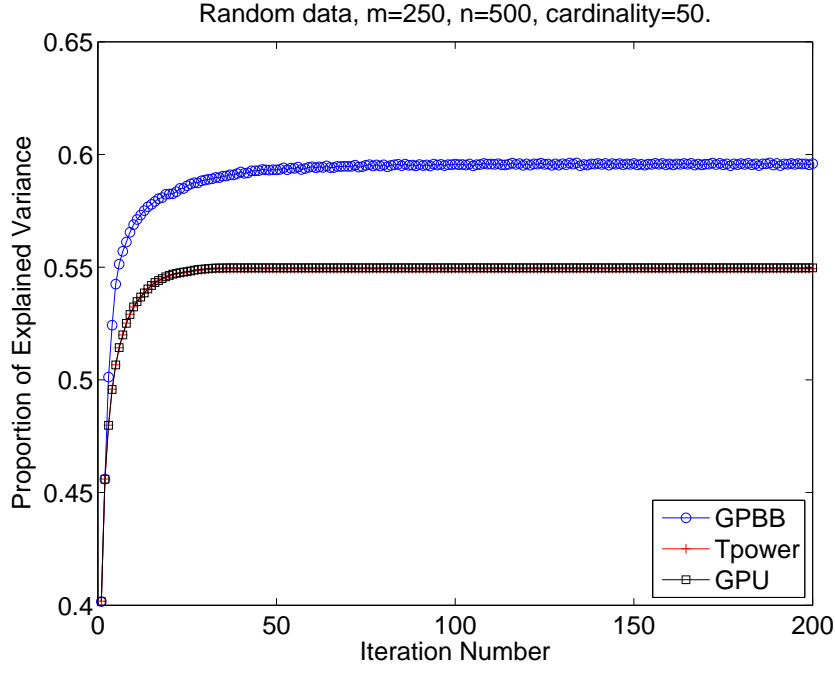


FIG. 4.3. Explained variance versus iteration number for cardinality 50 in the random data set.

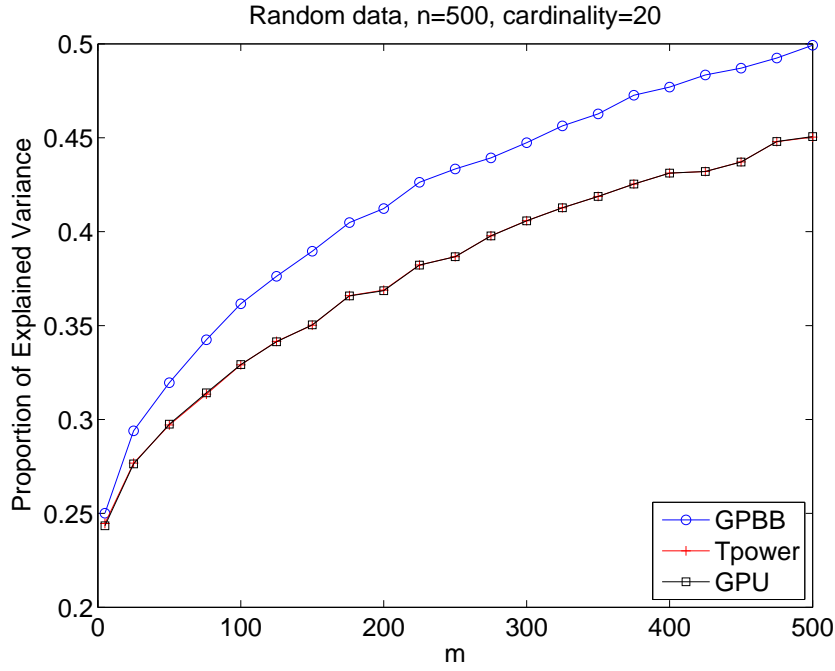


FIG. 4.4. Explained variance versus m for cardinality 20 in the random data set.

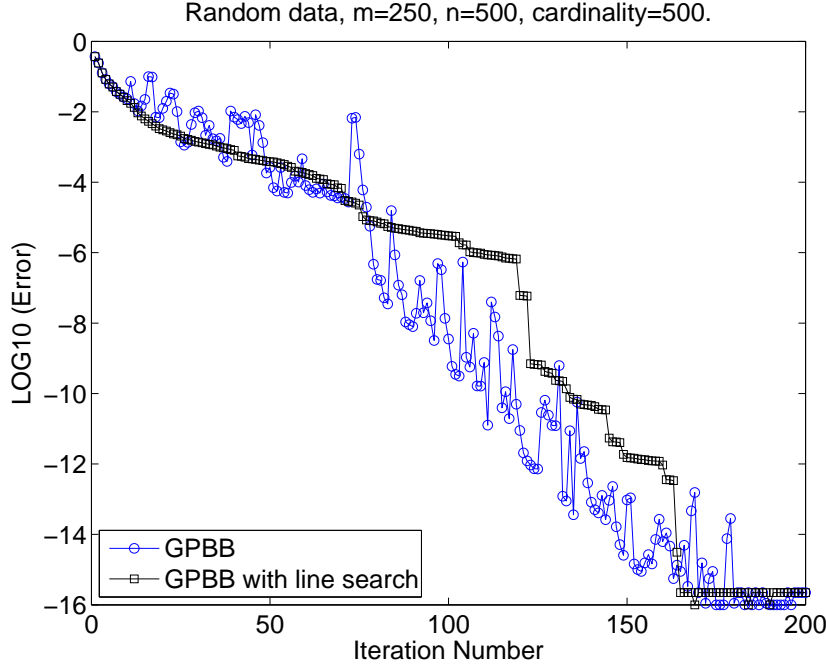


FIG. 4.5. A plot of the base 10 logarithm of the relative error versus iteration number for the random data set with $m = 250$ and cardinality $\kappa = 500$. The boxes correspond to the monotone algorithm which employs a line search, while the circles correspond to the nonmonotone algorithm with α_k given by the BB formula (3.13).

monotone GPBB with the monotone scheme corresponding to $\sigma = 0.25$. The error in the monotone algorithm converged to the computing precision around 10^{-16} in about the same number of iterations as the nonmonotone algorithm, however, the running time for the monotone algorithm was about 4 times larger since we may need to test several choices of α_k before generating a monotone iterate.

To compare with Gpower, we need to choose a value for γ . We first consider a simple case $m = 20, n = 20$ and we use the “default” seed in MATLAB to generate this matrix. The algorithms are used to extract the first principal component with $\kappa = 5$, and with γ tuned to achieve $\kappa = 5$. The results in Table 4.2 indicate that Gpower performed similar to Tpower, but not as well as GPBB.

TABLE 4.2
Simple random dataset

Method	Cardinality	Explained Variance
GPBB	$\kappa = 5$	0.8193
Tpower(ConGradU)	$\kappa = 5$	0.7913
Gpower _{l_1}	$\gamma = 0.18 (\Leftrightarrow \kappa = 5)$	0.7914
Gpower _{l_0}	$\gamma = 0.045 (\Leftrightarrow \kappa = 5)$	0.7914

In the next experiment, we consider 100 randomly generated matrices with $m = 250$ and $n = 500$, and with the parameter γ in Gpower chosen to achieve an average cardinality near 100 or 120. As seen in Table 4.3, Gpower _{l_0} achieves similar values for the proportion of the explained variance as Tpower, while Gpower _{l_1} achieves slightly better results and GPBB achieves the best results.

 TABLE 4.3
Random data set, $m = 250$, $n = 500$.

Method	Cardinality	Explained Variance
GPBB	$\kappa = 100$	0.7396
GPBB	$\kappa = 120$	0.7823
Tpower(ConGradU)	$\kappa = 100$	0.7106
Tpower(ConGradU)	$\kappa = 120$	0.7536
Gpower _{l_1}	$\gamma = 0.075$ (average $\kappa = 99$)	0.7288
Gpower _{l_1}	$\gamma = 0.0684$ (average $\kappa = 120$)	0.7679
Gpower _{l_0}	$\gamma = 0.0078$ (average $\kappa = 100$)	0.7129
Gpower _{l_0}	$\gamma = 0.0066$ (average $\kappa = 120$)	0.7557

Hollywood-2009 dataset, Densest k -subgraph (DkS). Given an undirected graph $G = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V} = \{1, 2, \dots, n\}$ and edge set \mathcal{E} , and given an integer $k \in [1, n]$, the densest k -subgraph (DkS) problem is to find a set of k vertices whose average degree in the subgraph induced by this set is as large as possible. Algorithms for finding DkS are useful tools for analyzing networks. Many techniques have been proposed for solving this problem including [5], [19], [26]. Mathematically, DkS is equivalent to a binary quadratic programming problem

$$(4.4) \quad \max\{\pi^T A \pi : \pi \in \mathbb{R}^n, \quad \pi \in \{0, 1\}^n, \quad \|\pi\|_0 = k\},$$

where A is the adjacency matrix of the graph; $a_{ij} = 1$ if $(i, j) \in \mathcal{E}$, while $a_{ij} = 0$ otherwise. We relax the constraints $\pi \in \{0, 1\}^n$ and $\|\pi\|_0 = k$ to

$\|\pi\| = \sqrt{k}$ and $\|\pi\|_0 \leq k$, and consider the following relaxed version of (4.4)

$$(4.5) \quad \max\{\pi^\top A \pi : \pi \in \mathbb{R}^n, \quad \|\pi\| = \sqrt{k}, \quad \|\pi\|_0 \leq k\}.$$

After a suitable scaling of π , this problem reduces to the sparse PCA problem (1.2).

Let us consider the Hollywood-2009 dataset [6, 7], which is associated with a graph whose vertices are actors in movies, and an edge joins two vertices whenever the associated actors appear in a movie together. The dataset can be downloaded from the following web site:

<http://law.di.unimi.it/datasets.php>

The adjacency matrix A is 1139905×1139905 . In order to apply Gpower to the relaxed problem, we first factored $A + cI$ into a product of the form $R^\top R$ using a Cholesky factorization, where $c > 0$ is taken large enough to make $A + cI$ positive definite. Here, R plays the role of the data matrix. However, one of the steps in the Gpower code updates the data matrix by a rank one matrix, and the rank one matrix caused the updated data matrix to exceed the 200 GB memory on the largest computer readily available for the experiments. Hence, this problem was only solved using Tpower and GPBB. Since the adjacency matrix requires less than 2 GB memory, it easily fit on our 8 GB computer.

In Table 4.4, we compare the density values $\pi^\top A \pi / k$ obtained by the algorithms. In addition, we also computed the largest eigenvalue λ of the adjacency matrix A , and give the ratio of the density to λ . Observe that in 2 of the 6 cases, GPBB obtained a significantly better value for the density when compared to Tpower, while in the other 4 cases, both algorithms converged to the same maximum.

TABLE 4.4
Hollywood data set.

Method	Cardinality	Density $\pi^\top A \pi / k$	Ratio $\frac{\pi^\top A \pi / k}{\lambda}$
GPBB	$k = 500$	379.40	0.1688
GPBB	$k = 600$	401.22	0.1785
GPBB	$k = 700$	593.24	0.2639
GPBB	$k = 800$	649.67	0.2891
GPBB	$k = 900$	700.38	0.3116
GPBB	$k = 1000$	745.95	0.3319
Tpower(ConGradU)	$k = 500$	190.11	0.0846
Tpower(ConGradU)	$k = 600$	401.21	0.1785
Tpower(ConGradU)	$k = 700$	436.53	0.1942
Tpower(ConGradU)	$k = 800$	649.67	0.2891
Tpower(ConGradU)	$k = 900$	700.44	0.3116
Tpower(ConGradU)	$k = 1000$	745.95	0.3319

5. Conclusion. The gradient projection algorithm was studied in the case where the constraint set Ω may be nonconvex, as it is in sparse principal component analysis. Each iteration of the gradient projection algorithm satisfied the condition $\nabla f(x_k)(x_{k+1} - x_k) \leq 0$. Moreover, if f is concave over $\text{conv}(\Omega)$, then $f(x_{k+1}) \leq f(x_k)$ for each k . When a subsequence of the iterates converge to x^* , we obtain in Theorem 2.6 the equality $\nabla f(x^*)(y^* - x^*) = 0$ for some $y^* \in P_\Omega(x^* - s^*g(x^*))$ where P_Ω projects a point onto Ω and s^* is the limiting step size. When Ω is convex, y^* is unique and the condition $\nabla f(x^*)(y^* - x^*) = 0$ is equivalent to the first-order necessary optimality condition at x^* for a local minimizer.

The approximate Newton algorithm with a positive Hessian approximation α_k , the iteration reduced to the projected gradient algorithm with step size $1/\alpha_k$. On the other hand, when $\alpha_k < 0$, as it is when the objective function is concave and the Hessian approximation is computed by the Barzilai-Borwein formula (3.13), the iteration amounts to taking a step along the positive gradient, and then moving as far away as possible while staying inside the feasible set. In numerical experiments based on sparse principal component analysis, the gradient projection algorithm with unit step size performed similar to both the truncated power method and the generalized power method. On the other hand, in some cases, the approximate Newton algorithm with a Barzilai-Borwein stepsize could converge much faster to a better objective function value than the other methods.

REFERENCES

- [1] A. A. ALIZADEH ET AL., *Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling*, Nature, 403 (2000), pp. 503–511.
- [2] J. BARZILAI AND J. M. BORWEIN, *Two point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.
- [3] AMIR BECK AND MARC TEBoulLE, *Mirror descent and nonlinear projected subgradient methods for convex optimization*, Operations Research Letters, 31 (2003), pp. 167–175.
- [4] D. P. BERTSEKAS, *Projected Newton methods for optimization problems with simple constraints*, SIAM J. Control Optim., 20 (1982), pp. 221–246.
- [5] A. BHASKARA, M. CHARIKAR, E. CHLAMTAC, U. FEIGE, AND A. VIJAYARAGHAVAN, *Detecting high log-densities: an $o(n^{1/4})$ approximation for densest k -subgraph*, in Proceedings of the forty-second ACM symposium on Theory of Computing, ACM, 2010, pp. 201–210.
- [6] PAOLO BOLDI, MARCO ROSA, MASSIMO SANTINI, AND SEBASTIANO VIGNA, *Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks*, in Proceedings of the 20th international conference on World Wide Web, ACM Press, 2011.
- [7] PAOLO BOLDI AND SEBASTIANO VIGNA, *The WebGraph framework I: Compression techniques*, in Proc. of the Thirteenth International World Wide Web Conference (WWW 2004), Manhattan, USA, 2004, ACM Press, pp. 595–601.
- [8] JORGE CADIMA AND IAN T JOLLIFFE, *Loading and correlations in the interpretation of principle components*, Journal of Applied Statistics, 22 (1995), pp. 203–214.
- [9] KENNETH L CLARKSON, *Coresets, sparse greedy approximation, and the Frank-Wolfe*

- algorithm*, ACM Transactions on Algorithms (TALG), 6 (2010), p. 63.
- [10] ALEXANDRE D'ASPREMONT, FRANCIS BACH, AND LAURENT EL GHAOUI, *Optimal solutions for sparse principal component analysis*, The Journal of Machine Learning Research, 9 (2008), pp. 1269–1294.
 - [11] M. FRANK AND P. WOLFE, *An algorithm for quadratic programming*, Naval Research Logistics Quarterly, 3 (1956), pp. 95–110.
 - [12] W. W. HAGER AND H. ZHANG, *A new active set algorithm for box constrained optimization*, SIAM J. Optim., 17 (2006), pp. 526–557.
 - [13] E. HAZAN AND S. KALE, *Projection-free online learning*, in Proceedings of the 29th International Conference on Machine Learning, J. Langford and J. Pineau, eds., Omnipress, 2012, pp. 521–528.
 - [14] MARTIN JAGGI, *Revisiting Frank-Wolfe: Projection-free sparse convex optimization*, in Proceedings of the 30th International Conference on Machine Learning, S. Dasgupta and D. McAllester, eds., vol. 28, 2013, pp. 427–435.
 - [15] J. JEFFERS, *Two case studies in the application of principal components*, Applied Statistics, 16 (1967), pp. 225–236.
 - [16] R. JENATTON, G. OBOZINSKI, AND F. BACH, *Structured sparse principal component analysis*, in International Conference on Artificial Intelligence and Statistics (AISTATS), 2010.
 - [17] IAN T JOLLIFFE, NICKOLAY T TRENDAFILOV, AND MUDASSIR UDDIN, *A modified principal component technique based on the LASSO*, Journal of Computational and Graphical Statistics, 12 (2003), pp. 531–547.
 - [18] MICHEL JOURNÉE, YURI NESTEROV, PETER RICHTÁRIK, AND RODOLPHE SEPULCHRE, *Generalized power method for sparse principal component analysis*, The Journal of Machine Learning Research, 11 (2010), pp. 517–553.
 - [19] S. KHULLER AND B. SAHA, *On finding dense subgraphs*, in Automata, Languages and Programming, Springer, 2009, pp. 597–608.
 - [20] SIMON LACOSTE-JULIEN, MARTIN JAGGI, MARK SCHMIDT, AND PATRICK PLETSCHER, *Block-coordinate Frank-Wolfe optimization for structural SVMs*, in Proceedings of the 30th International Conference on Machine Learning, S. Dasgupta and D. McAllester, eds., vol. 28, 2013, pp. 53–61.
 - [21] RONNY LUSS AND MARC TEBoulLE, *Convex approximations to sparse PCA via Lagrangian duality*, Operations Research Letters, 39 (2011), pp. 57–61.
 - [22] ———, *Conditional gradient algorithms for rank-one matrix approximations with a sparsity constraint*, SIAM Review, 55 (2013), pp. 65–98.
 - [23] SRIDHAR RAMASWAMY, PABLO TAMAYO, RYAN RIFKIN, SAYAN MUKHERJEE, CHEN-HSIANG YEANG, MICHAEL ANGELO, CHRISTINE LADD, MICHAEL REICH, EVA LATULIPPE, JILL P MESIROV, ET AL., *Multiclass cancer diagnosis using tumor gene expression signatures*, Proceedings of the National Academy of Sciences, 98 (2001), pp. 15149–15154.
 - [24] R. T. ROCKAFELLAR, *Convex analysis*, Princeton Univ. Press, 1970.
 - [25] BHARATH K SRIPERUMBUDUR, DAVID A TORRES, AND GERT RG LANCKRIET, *A majorization-minimization approach to the sparse generalized eigenvalue problem*, Machine Learning, 85 (2011), pp. 3–39.
 - [26] Y. YE AND J. ZHANG, *Approximation of dense- $n/2$ -subgraph and the complement of min-bisection*, Journal of Global Optimization, 25 (2003), pp. 55–73.
 - [27] XIAO-TONG YUAN AND TONG ZHANG, *Truncated power method for sparse eigenvalue problems*, The Journal of Machine Learning Research, 14 (2013), pp. 899–925.
 - [28] HUI ZOU, TREVOR HASTIE, AND ROBERT TIBSHIRANI, *Sparse principal component analysis*, Journal of Computational and Graphical Statistics, 15 (2006), pp. 265–286.